Autenticação de Usuário

PROGRAMAÇÃO WEB 1

Objetivos de Aprendizagem

- Exemplificar o uso de autenticação com Passport

Agenda

“O Passport é uma middleware de autenticação para Node.js. É extremamente flexível e modular podendo ser aplicado a qualquer aplicação com Express. Um conjunto completo de estratégias para autenticação pode ser utilizada, desde usuário/senha, Google, Facebook, Twitter, dentre mais de 500 opções”

https://www.passportjs.org/

Autenticação

- A autenticação refere-se a restringir acesso a aplicação mediante identificação do usuário
- A partir da identificação, áreas personalizadas da aplicação podem ser disponibilizadas ao usuário
- A estratégia de autenticação mais simples utiliza usuário, senha e um BD próprio da aplicação

Usuário e Senha

- Tutorial ToDo App
  - Aplicação de exemplo para criar uma lista de a fazeres de um usuário
  - https://www.passportjs.org/tutorials/password/
- No repositório acima a versão inicial da aplicação é disponibilizada

Funcionalidades ToDo App

- Login *prompt*
- Verificação de senha
- Estabelecimento de sessão
- Logout
- Criação de conta

Login Prompt (*Sign in*)

- Permite aos usuários entrar com suas credenciais (usuário/senha) para ter iniciar uma sessão

1. Criar o arquivo `auth.js` no diretório `routes`

routes > auth.js > ...

```
1  var express = require('express');
2  var router = express.Router();
3
4  router.get('/login', function(req, res) {
5    res.render('login');
6  });
7
8  module.exports = router;
```

Login Prompt

2. Adicionar a rota `/login` em `app.js`

```
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');

var indexRouter = require('./routes/index');
var authRouter = require('./routes/auth');

var app = express();

app.locals.pluralize = require('pluralize');
```

app.js > ...

```
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/', authRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
```

Login Prompt

3. Adicionar formulário em `login.ejs` no diretório `views`

views > login.ejs

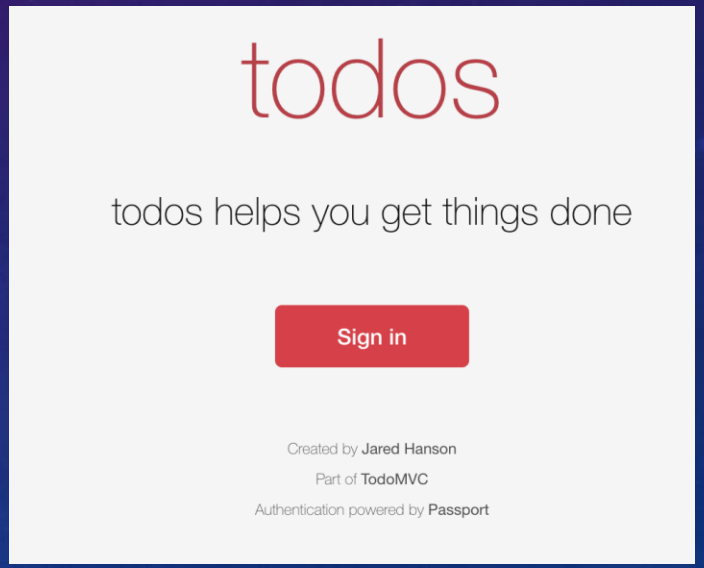
```
13        <h3>todos</h3>
14        <h1>Sign in</h1>
15
16        <form action="/login/password" method="post">
17            <section>
18                <label for="username">Username</label>
19                <input id="username" name="username" type="text" autocomplete="username" required autofocus>
20            </section>
21            <section>
22                <label for="current-password">Password</label>
23                <input id="current-password" name="password" type="password" autocomplete="current-password"
    required>
24            </section>
25            <button type="submit">Sign in</button>
26        </form>
27        <hr>
28        <p class="help">Don't have an account? <a href="/signup">Sign up</a></p>
29    </section>
```

Login Prompt

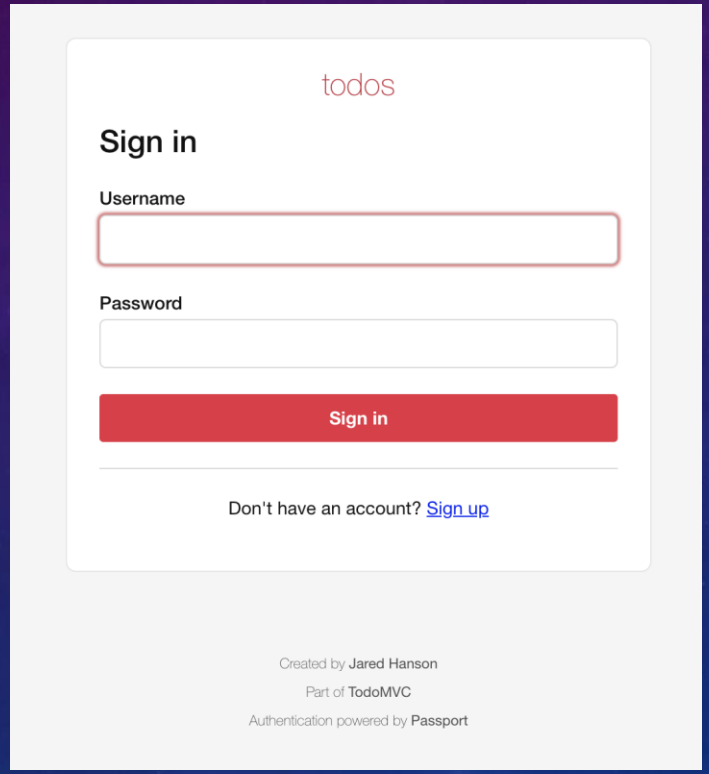
4. Iniciar a aplicação

```
npm start
```

5. Clicar em Sign in



Login Prompt



Verificação de Senha

- Agora será feita a parte da verificação da senha

1. Instalar as dependências

```
$ npm install passport

$ npm install passport-local
```

Verificação de Senha

2. Configurar o `passport` em `auth.js`

routes > auth.js > ...

```
1  var express = require('express');
2  var router = express.Router();
3
4  var passport = require('passport');
5  var LocalStrategy = require('passport-local');
6  var crypto = require('crypto');
7  var db = require('../db');
8
```

Verificação de Senha

3. Configurar `LocalStrategy` em `auth.js`

routes > auth.js > ...

```
7  var db = require('../db');
8
9  passport.use(new LocalStrategy(function verify(username, password, cb) {
10   db.get('SELECT * FROM users WHERE username = ?', [ username ], function(err, row) {
11     if (err) { return cb(err); }
12     if (!row) { return cb(null, false, { message: 'Incorrect username or password.' });
13
14     crypto.pbkdf2(password, row.salt, 310000, 32, 'sha256', function(err, hashedPassword) {
15       if (err) { return cb(err); }
16       if (!crypto.timingSafeEqual(row.hashed_password, hashedPassword)) {
17         return cb(null, false, { message: 'Incorrect username or password.' });
18       }
19       return cb(null, row);
20     });
21   });
22 }));
23
```

Verificação de Senha

4. Adicionar o *middleware* que realiza a autenticação em `auth.js`

routes > auth.js > ...

```
36 router.get('/login', function(req, res) {
37   res.render('login');
38 });
39
40 router.post('/login/password', passport.authenticate('local', {
41   successRedirect: '/',
42   failureRedirect: '/login'
43 }));
44
```

Verificação de Senha

5. Reiniciar a aplicação

```
npm start
```

6. Inserir usuário/senha

- Observar que logins incorretos levam a página de login novamente

Verificação de Senha

7. Inserir usuário cadastrado

```
Username: alice

Password: letmein
```

8. Erro

Login sessions require session support. Did you forget to u

```
Error: Login sessions require session support. Did you forget to use `express-session` middleware?
    at SessionManager.logIn (/home/runner/todos-express-starter/node_modules/passport/lib/sessionmanag
    at req.login.req.logIn (/home/runner/todos-express-starter/node_modules/passport/lib/http/request.
    at strategy.success (/home/runner/todos-express-starter/node_modules/passport/lib/middleware/authe
    at verified (/home/runner/todos-express-starter/node_modules/passport-local/lib/strategy.js:83:10)
    at PBKDF2Job.<anonymous> (/home/runner/todos-express-starter/routes/auth.js:19:14)
    at job.ondone (node:internal/crypto/pbkdf2:55:12)
```

Estabelecimento de Sessão

- Depois que o usuário e a senha são autenticados, é necessário estabelecer uma sessão
- A sessão guarda os dados do usuário a medida que navega na aplicação
- Para isso, são necessárias mais duas dependências:
  - `express-session`
  - `connect-sqlite3`

Estabelecimento de Sessão

1. Instalar dependências

```
npm install express-session

npm install connect-sqlite3
```

2. Atualizar o `app.js`

```
6  var cookieParser = require('cookie-parser');
7  var logger = require('morgan');
8  var passport = require('passport');
9  var session = require('express-session');
10 var SQLiteStore = require('connect-sqlite3')(session)
11
12 var indexRouter = require('./routes/index');
13 var authRouter = require('./routes/auth');
14
```

Estabelecimento de Sessão

3. Configuração de sessão

app.js > ...

```
27  app.use(express.static(path.join(__dirname, 'public')));
28
29  app.use(session({
30    secret: 'keyboard cat',
31    resave: false,
32    saveUninitialized: false,
33    store: new SQLiteStore({ db: 'sessions.db', dir: './var/db' })
34  }));
35  app.use(passport.authenticate('session'));
36
37  app.use('/', indexRouter);
```

Estabelecimento de Sessão

4. Adicionar novas funções em `auth.js`

routes > auth.js > verify > passport.deserializeUser() callback > process.nextTick() callback

```
16      if (!crypto.timingSafeEqual(row.hashed_password, hashedPassword)) {
17        return cb(null, false, { message: 'Incorrect username or password.' });
18      }
19      return cb(null, row);
20    });
21  });
22
23  passport.serializeUser(function(user, cb) {
24    process.nextTick(function() {
25      cb(null, { id: user.id, username: user.username });
26    });
27  });
28
29  passport.deserializeUser(function(user, cb) {
30    process.nextTick(function() {
31      return cb(null, user);
32    });
33  });
34  }));
```

Logout

- Realiza a finalização da sessão

Logout

1. Criar o *middleware* que realiza logout em `auth.js`

routes > auth.js > ...

```
40 router.post('/login/password', passport.authenticate('local', {
41   successRedirect: '/',
42   failureRedirect: '/login'
43 }));
44
45 router.post('/logout', function(req, res, next) {
46   req.logout(function(err) {
47     if (err) { return next(err); }
48     res.redirect('/');
49   });
50 });
51
52 module.exports = router;
```

Criação de Conta

1. Adicionar o *middleware* para `/signup` em `auth.js`

routes > auth.js > ...

```
46    req.logout(function(err) {
47      if (err) { return next(err); }
48      res.redirect('/');
49    });
50  });
51
52  router.get('/signup', function(req, res, next) {
53    res.render('signup');
54  });
55
56  module.exports = router;
```

Criação de Conta

2. Adicionar o formulário de criação de contas em `signup.ejs` (views)

views > signup.ejs

```
12    <section class="prompt">
13      <h3>todos</h3>
14      <h1>Sign up</h1>
15
16      <form action="/signup" method="post">
17          <section>
18              <label for="username">Username</label>
19              <input id="username" name="username" type="text" autocomplete="username" required>
20          </section>
21          <section>
22              <label for="new-password">Password</label>
23              <input id="new-password" name="password" type="password" autocomplete="new-password" required>
24          </section>
25          <button type="submit">Sign up</button>
26      </form>
27
28      <hr>
29      <p class="help">Already have an account? <a href="/login">Sign in</a></p>
30    </section>
```

Criação de Conta

3. Adicionar nova rota para a criação de contas em `auth.ejs`

routes > auth.js > ...

```
55
56  router.post('/signup', function(req, res, next) {
57    var salt = crypto.randomBytes(16);
58    crypto.pbkdf2(req.body.password, salt, 310000, 32, 'sha256', function(err, hashedPassword) {
59      if (err) { return next(err); }
60      db.run('INSERT INTO users (username, hashed_password, salt) VALUES (?, ?, ?)', [
61        req.body.username,
62        hashedPassword,
63        salt
64      ], function(err) {
65        if (err) { return next(err); }
66        var user = {
67          id: this.lastID,
68          username: req.body.username
69        };
70        req.login(user, function(err) {
71          if (err) { return next(err); }
72          res.redirect('/');
73        });
74      });
75    });
76  });
```

Limitando o Acesso

- Como restringir o acesso apenas aos usuários autenticados?

1. Em `app.js` criar a função `isLoggedIn()`

app.js > isLoggedIn

```
36  app.use(passport.authenticate('session'));
37
38  function isLoggedIn(req, res, next) {
39    if (req.isAuthenticated()) {
40      return next();
41    }
42    res.redirect('/login');
43  }
44
45  app.use('/', indexRouter);
46  app.use('/', authRouter);
```

Limitando o Acesso

- Agora basta aplicar a função `isLoggedIn()` às rotas que se deseja restringir apenas aos usuários que fizerem login
- As rotas públicas ficam inalteradas

2. Em `app.js` atualizar as rotas a serem protegidas

app.js > isLoggedIn

```
8  var passport = require('passport');
9  var session = require('express-session');
10 var SQLiteStore = require('connect-sqlite3')(sess
11
12 var indexRouter = require('./routes/index');
13 var authRouter = require('./routes/auth');
14 var loggedRouter = require('./routes/logged');
```

app.js > isLoggedIn

```
34   store: new SQLiteStore({ db: 'sessions
35 }));
36 app.use(passport.authenticate('session')
37
38 function isLoggedIn(req, res, next) {
39   if (req.isAuthenticated()) {
40     return next();
41   }
42   res.redirect('/login');
43 }
44
45 app.use('/', indexRouter);
46 app.use('/', authRouter);
47 app.use('/', isLoggedIn, loggedRouter);
```

Limitando o Acesso

- Agora basta aplicar a função `isLoggedIn()` às rotas que se deseja restringir apenas aos usuários que fizerem login
- As rotas públicas ficam inalteradas

3. Exemplo de página protegida

routes > logged.js > ...

```
1  var router = require('express').Router();
2
3  router.get('/only', function(req, res) {
4    res.send('I am authenticated!!!');
5  });
6
7  module.exports = router;
```

Referências

- https://www.passportjs.org/packages/
- https://www.sqlitetutorial.net/sqlite-nodejs/query/
- https://www.passportjs.org/concepts/authentication/password/
- https://www.geeksforgeeks.org/node-js-crypto-pbkdf2-method/
- https://www.geeksforgeeks.org/node-js-crypto-timingsafeequal-function/